# An Object Code Compression Approach to Embedded Processors

Yukihiro Yoshida, Bao-Yu Song, Hiroyuki Okuhata,
Takao Onoye, and Isao Shirakawa
**Dept. Information Systems Engineering, Osaka University**
Phone: +81(6)879-7808, FAX: +81(6)875-5902
*e-mail: {yoshida, song, okuhata, onoye, shirakawa }@ise.eng.osaka-u.ac.jp*

## ABSTRACT

A low-power processor architecture is described dedicatedly for embedded application programs by means of an object code compression approach. This approach unifies application instructions existing in the embedded program and assigns compressed object codes to such an instruction. An instruction decompressor is constructed to generate the object code from a compressed object code (pseudo code) input. A single-chip implementation of this decompressor together with a processor core can effectively reduce the bandwidth required for the I/O interface. To demonstrate the practicability of the proposed approach, experiments are applied to an embedded processor ARM610 which attains 62.5% decompression, hence 42.3% of the power consumption of instruction memory can be reduced.

## 1 INTRODUCTION

Recently the market of embedded processors is growing rapidly according to the increased different demands for the applications to mobile computing, consumer electronics [1,2]. Although 4-, 8-, and 16-bit microprocessors have dominated conventionally the embedded processor market, the advanced applications require not only much higher performance but also more extensive versatility. In addition, with the recent advance of the fabrication technology, many of ARM and SPARC processors can be soon replaced by another one through renewal of fabrication process, which has space enough to admit a variety of additional functions.

The low-power consumption is one of the main subjects for the embedded processors to be used for portable peripherals such as PDAs (Personal Digital Assistants) and PHSs (Personal Handy-phone Systems) especially in terms of enhancing the battery lives. Thus the low-power design has become the key factor for embedded applications, a number of VLSI architectures have been proposed [4,3,5,6,7,8] with the main focus put on low supply voltage, transfer gates, gated clocks, etc. However, most of these architectures are based on the circuit-level power saving. This means that there still remains much room for reducing power consumption through system-level power saving mechanisms, such as reduction of external memory size, slowing down of drive speed of I/O interfaces, etc.

Generally the embedded application programs use a subset of the instruction set provided by a processor, as can be often observed in CISC (Complex Instruction Set Computer) processors as well as in RISC (Reduced Instruction Set Computer) processors. Hence, in this paper, a new approach is devised for the system-level low-power consumption, which is to reduce the bandwidth (i.e. the memory accessing speed and/or the instruction bit width) required for the I/O interface by means of compressing the object codes for specific embedded application programs. The system-level power saving is to be achieved by reducing the dissipated power of the external memory. In addition, seeing that an embedded processor can be easily integrated on a single chip with the use of such add-on circuits, the object code compression can be effectively applied to embedded application programs on embedded processors.

## 2 OBJECT CODE COMPRESSION

The overview of the embedded processor to be discussed is outlined in Fig. 1, where specific functional on-chip memories etc. are integrated together with the processor core in a single chip. It should be stressed that the processor is usually accompanied with the external memory to contain object code and data.

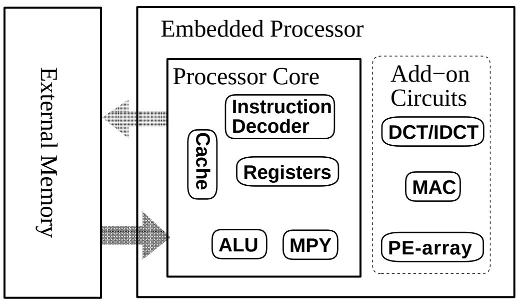


Figure 1: Overview of low-power enhanced embedded processor.

The approach to be described here intends to reduce the power consumption from the viewpoint of the following factors of embedded processors.

- The main issue is to reduce the power consumption of the total system which is composed of the external memory, interfaces, etc.
- The size of each program which runs on the processor is relatively small.
- Additional circuits can be easily integrated into a chip.

Taking account of these factors, the power consumption of the external memory can be regarded as comparable to that of the embedded processor, that is, the reduction of power consumption of the external memory greatly affects the system level power saving of the total system.

Now, it should be remarked that a program generally uses only a small part of the instruction set provided by the processor, and moreover there are much duplication of instructions in the program. In other words, a set of much shorter instructions may execute the functions required for the program. An instruction decompressor is synthesized so as to generate object codes from such shorter instructions, which can be easily integrated into the embedded processor.

Based on these considerations, an object compression scheme is constructed as outlined below.

**1°** Given an embedded program of $m$-bit width, trace it to get a set $G$ $(:= \{i | i = 1, 2, ..., n\})$ of instructions without duplication.

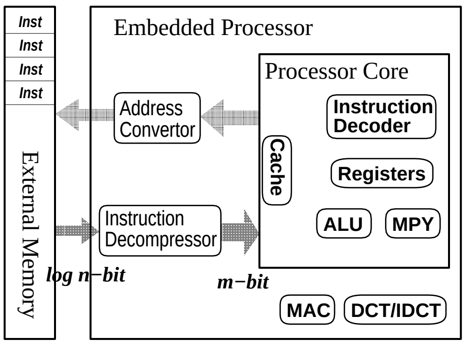
**2°** Assign a number $i$ to each distinct instruction of $G$ as a log $n$-bit code.

**3°** Construct a table to transform each pseudo code to an $m$-bit-wide instruction, and implement it in the instruction decompressor.

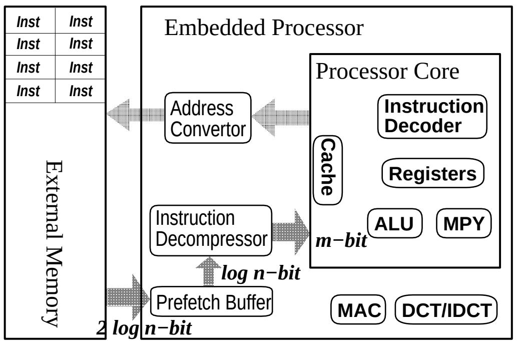
In this way, the I/O bandwidth between the processor and the external memory is reduced. If the external memory contains the pseudo code itself, as illustrated in Fig.2(a), the bitwidth accessing to the memory can be reduced to (log $n$)/$m$. If a few pseudo codes are grouped into an $m$-bit code, as indicated in Fig. 2(b), a slow speed memory can be employed. In either case, the power consumption of the system can be greatly reduced.

The transform table of address/instruction can be easily implmeneted by using ROM as shown in Fig.3. With the recent advance of the fablication technology the memory unit can be intagrated on a processor die [9,10, 11], and hence the transform table can be incorporated into the embedded processor.

The so-called Thumb instruction set [12] for an ARM processor core can reduce the I/O bandwidth for embedded application programs. This scheme, however, introduces a new instruction set composed of 36 16-bit operation codes drawn from the standard 32-bit ARM instruction set, for which additional software tools (i.e. compiler, assembler, linker, etc.) should be developed. On the contrary, our approach necessitates only simple software tools for tracing of programs, and sorting, unifying, and numbering of instructions. Eventually, our approach need neither devise a new instruction set nor develop elaborate software tools.



(a) organization with narrow width external memory



(b) organization with low speed external memory

Figure 2: Overview of low power enhanced embedded processors.

## 3 FULLCODE/SUBCODE COMPRESSION

Let us denote by the *fullcode compression* scheme the one stated above, which is performed by a series of steps **1°**, **2°**, and **3°**.

Consider that in the instruction format, registers and flags are assigned to fixed positions, and let us now introduce another *sub code compression* scheme, in which operation codes can be compressed similarly, with the codes for registers and flags bypassing directly to the processor core, as shown in Fig.4.

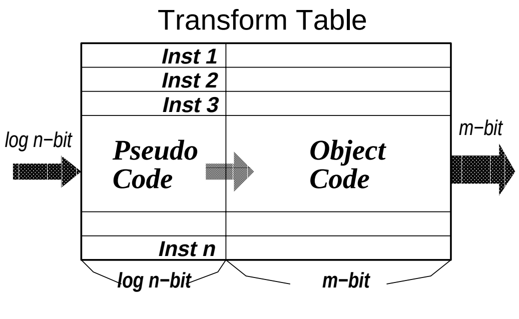


Figure 3: Instruction decompressor for full code compression.

Generally, the power dissipation of a memory unit depends on the area occupancy. Hence, the power reduction ratio $P_{f/o}$ of the instruction memory by the full code com-

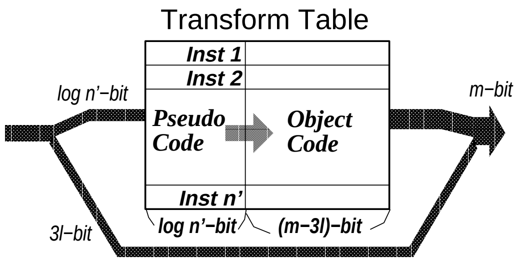


Figure 4: Instruction decompressor for subcode compression.

pression can be defined by

$$P_{f/o} = 1 - \frac{N\lceil \log n \rceil + knm}{Nm}, \tag{1}$$

where $N$, $m$, and $n$ indicate the number of instrcutions for original program, the bitwidth of original instruction, the number of compressed instructions, respectively. In addition, $k$ indicates the power dissipation ratio of the on-chip memory to the external memory, and actually $k$ is in the range of $0.5 \leq k \leq 0.7$.

In the same manner with this, the power reduction ratio $P_{s_i/o}$ of the instruction memory by the subcode compression, in which $i$ register operands are passed directly to the processor core, can be defined by

$$P_{s_i/o} = 1 - \frac{N(\lceil \log n' \rceil + il) + kn'(m - il)}{Nm}, \tag{2}$$

where, $l$ and $n'$ indicate the bitwidth of each register operands and the number of compressed instructions, respectively.

## 4 EXPERIMENTAL RESULTS

The proposed compression scheme has been applied to an ARM610 processor core. Fig. 5 outlines the process flow of our object code compression system. The existing software development kit (*ARMsdt*) is employed for tracing instructions, and the *Peal* for the data processing. In the fullcode compression, only object codes are input to this system, whereas in the subcode compression, in addition to the object codes, the instruction formats are also input to this system, as indicated in Fig. 5. This system outputs either ROM generated by memory compiler of *COMPASS Design Navigator* or logic cells generated by VHDL synthesizer.

Several experiments have been attempted by using a number of embedded programs, such as the so-called Dhrystone benchmark (dhrystone) and an PDA operating system $\mu$ITRON (uitron), in order to observe the practicability of the proposed scheme.

Table 1 shows a part of experimental results of the fullcode and subcode compressions. As summarized in Fig. 6, the instruction trace outputs object codes, frequencies of their appearance together with percentage. Fig. 7 illustrates the transform table from the pseudo code to the ARM instruction. The power reduction ratio determined by equations (1) and (2) are summarized in Table 2, where the coefficient $k$ is set to 0.7.

As for the PDA operating system $\mu$ITRON, the number of instructions by the fullcode compression is 4,026, each assigned by 12-bit pseudo code. The compression ratio of

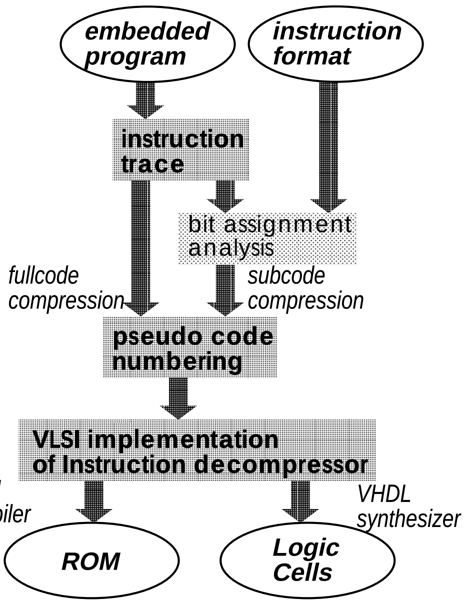
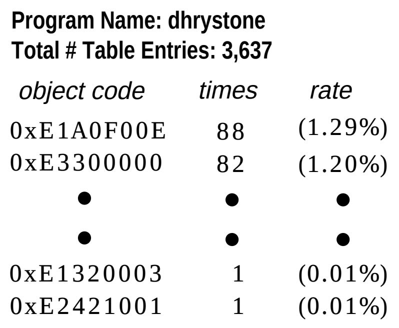


Figure 5: Process flow of object code compression.


Program Name: dhrystone
Total # Table Entries: 3,637

| object code | times | rate |
|---|---|---|
| 0xE1A0F00E | 88 | (1.29%) |
| 0xE3300000 | 82 | (1.20%) |
| ● | ● | ● |
| ● | ● | ● |
| 0xE1320003 | 1 | (0.01%) |
| 0xE2421001 | 1 | (0.01%) |

Figure 6: Instruction tracing result for Dhrystone.

bitwidth from 32 to 12 is 62.5%, and the power reduction ratio of the instruction memory is 42.3%. Table 3 indicates the implementation results of instruction decompressor for $\mu$ITRON. It should be remarkable that the logic cells generation is especially effective in the case of subcode compression.

The important benefit is that the system can be constructed with the use of only one 16-bit external memory chip. Now that only 8- or 16-bit memory chips are available, in order to feed 32-bit instructions to the processor directly from the external memory, it should be necessary to use four 8-bit memory chips or two 16-bit memory chips, even though a single chip could sufficiently supply the memory capacity.

## 5 CONCLUSION

This paper has described a low-power consumption scheme dedicated to embedded processors by object code compression. Adopting pseudo instructions in object codes of embedded programs and add-on circuits, system level power

Table 1: Experimental results.

| Program | Original Size | Full code | Sub code | |
|---|---|---|---|---|
| | | | $i = 2$ | $i = 3$ |
| dhrystone | 6,733 | 3, 637 | 1, 992 | 1, 449 |
| uitron | 13, 970 | 4, 026 | 1, 430 | 1, 140 |
| armasm | 41, 506 | 14, 563 | 6, 269 | 4, 544 |
| armcc | 117, 549 | 37, 095 | 14, 505 | 9, 591 |
| armlib | 9, 136 | 4, 852 | 2, 667 | 1, 957 |
| armlink | 29, 050 | 12, 111 | 5, 844 | 4, 206 |
| armmake | 10, 405 | 5, 326 | 2, 940 | 2, 117 |
| armsd | 93, 092 | 28, 572 | 11, 377 | 7, 494 |
| armtools(*) | 300, 738 | 68, 418 | 23, 166 | 14, 384 |
| runimage | 124, 442 | 31, 587 | 9, 157 | 5, 337 |
| awrender | 13, 500 | 6, 985 | 3, 562 | 2, 602 |
| imageext | 4, 038 | 2, 295 | 1, 528 | 1, 208 |
| fontdraw | 2, 783 | 1, 831 | 1, 160 | 941 |

(*) armasm, armcc, armlib, armlink, armmake, armsd

```
0000 0000 0000 ─►1110 0001 1010 0000 1111 0000 0000 1110 #(0xE1A0F00E)
0000 0000 0001 ─►1110 0011 0011 0000 0000 0000 0000 0000 #(0xE3300000)
      ●                            ●
      ●                            ●
1110 0011 0100 ─►1110 0010 0100 0010 0001 0000 0000 0001 #(0xE2421001)
  pseudo code                 object code
```

Figure 7: Transform table from pseudo code to ARM instruction.

consumption is saved from the viewpoint of processors and external memory.

## ACKNOWLEDGEMENTS

The authors are grateful to Dr. T. Kawata and Dr. N. Kubo, SHARP Corporation, for their invaluable and continuous supports. The authors also thank to Mr. T. Brown, ARM Ltd., for providing ARM program data.

## REFERENCES

[1] A. Someren and C. Atack: "The ARM RISC Chip", *Addison-Wesley Publishing Company*, 1993.

[2] M. Dolle and M. Schlett: "A cost-effective RISC/DSP micropro cessor for embedded systems", *IEEE Micro*, vol. 15, no. 5, pp. 32–40, Oct. 1995.

[3] S. Malhi and P. Chatterjee: "1V microsystems– Scaling on schedule for personal communications", *IEEE Circuits and Devices*, vol. 10, no. 2, pp. 13–17, Mar. 1994.

[4] Y. Okaguro: "Design of a low-power RISC processor for embedded applications", *Technical Report of IEICE*, ICD95-60, Jun. 1995 (in Japanese).

[5] T. Enomoto: "Low-power CMOS and GaAs digital design for multimedia LSIs", *Technical Report of IEICE*, ICD95-69, Aug. 1995 (in Japanese).

[6] S. Horiguchi, T. Tsukahara, and H. Fukuda: "Low-power LSI circuit technologies for portable terminal equipment", *IEICE Trans. Electronics*, vol. E78-C, no. 12, pp. 1655–1667, Dec. 1995.

Table 2: Power reduction ratio of instruction memory.

| Program | Full code | Sub code | |
|---|---|---|---|
| | $P_{f/o}$ | $P_{s_2/o}$ | $P_{s_3/o}$ |
| dhrystone | 0. 2470 | 0. 2509 | 0. 1871 |
| uitron | 0. 4233 | 0. 3525 | 0. 2455 |
| armasm | 0. 3169 | 0. 2645 | 0. 1709 |
| armcc | 0. 2791 | 0. 2477 | 0. 1518 |
| armlib | 0. 2220 | 0. 2217 | 0. 1875 |
| armlink | 0. 2707 | 0. 2381 | 0. 1554 |
| armmake | 0. 2354 | 0. 2267 | 0. 1610 |
| armsd | 0. 3164 | 0. 2483 | 0. 1835 |
| tools | 0. 3095 | 0. 2408 | 0. 1666 |
| runimage | 0. 3536 | 0. 2739 | 0. 2000 |
| awrender | 0. 2316 | 0. 2365 | 0. 1657 |
| imageext | 0. 2272 | 0. 2076 | 0. 1504 |
| fontdraw | 0. 1957 | 0. 1874 | 0. 1646 |
| fontdraw | 0. 1957 | 0. 1874 | 0. 1646 |

Table 3: Implementation results of instruction decompressor for $\mu$ITRON

| | Full code ($mm^2$) | Sub code ($mm^2$) | |
|---|---|---|---|
| | | $i = 2$ | $i = 3$ |
| ROM | 0. 983 | 0. 364 | 0. 270 |
| Logic cells | 1. 442 | 0. 268 | 0. 190 |

0. 35$\mu m$ CMOS triple-metal technology

[7] S. Shigematsu, S. Mutoh, and Y. Matsuya: "Power management technique for 1-V LSIs using embedded processor", in *Proc. IEEE Custom Integrated Circuits Conference*, pp. 111-114, May 1996.

[8] G. C. Cardarilli, M. Salmeri, A. Salsano, and O. Simonelli: "Bus architecture for low-power VLSI digital circuits", in *Proc. IEEE Int'l Symp. Circuits and Systems*, pp. 4. 21–4. 24, May 1996.

[9] S. Iwata, T. Shimizu, J. Korematu, K. Dosaka, H. Tsubota, and K. Saitoh: "Porformance evaluation of a micropro cessor with on-chip DRAM and high bandwidth internal bus", in *Proc. IEEE Custom Integrated Circuits Conference*, pp. 269-272, May 1996.

[10] T. Shimizu, J. Korematu, M. Satou, H. Kondo, S. Iwata, K. Sawai, N. Okumura, K. Ishimi, Y. Nakamoto, M. Kumanoya, K. Dosaka, A. Yamazaki, Y. Ajioka, H. Tsubota, Y. Nonomura, T. Urabe, J. Hinata, and K. Saitoh: "A multimedia 32b RISC micropro cessor with 16Mb DRAM, in *IEEE ISSCC Digest of Technical Papers*, FP13. 4, Feb. 1996.

[11] Y. Yamagata, T. Ishibashi, Y. Sano, Y. Koga, M. Yoshida, and A. Sugo: "32-bit RISC micro controller V853", *NEC Technical Journal*, vol. 49, no. 3, pp. 55–60, Apr. 1996.

[12] S. Segars, K. Clarke, and L. Goudge: "Embedded control problems, Thumb and the ARM7TDMI", *IEEE Micro*, vol. 15, no. 5, pp. 22–30, Oct. 1995.